



TITLE:

ワークステーション上でのマイクロプログラムPrologインタプリタの構成について(知識ベースとデータベースの統合化に関する研究)

AUTHOR(S):

堀切, 和典; 吉田, 雄二; 福村, 晃夫

---

CITATION:

堀切, 和典 ...[et al]. ワークステーション上でのマイクロプログラムPrologインタプリタの構成について(知識ベースとデータベースの統合化に関する研究). 数理解析研究所講究録 1986, 593: 168-182

ISSUE DATE:

1986-06

URL:

<http://hdl.handle.net/2433/99505>

RIGHT:

## ワークステーション上でのマイクロプログラム P r o l o g インタプリタの構成について

名大工学部 堀切 和典 (Kazunori Horikiri)

名大工学部 吉田 雄二 (Yuuji Yoshida)

名大工学部 福村 晃夫 (Teruo Fukumura)

### 1. はじめに

我々は P r o l o g 処理系の効率を高めるために処理系で基本となるユニフィケーション, バックトラッキングなどの操作をマイクロプログラムで記述した P r o l o g インタプリタを高性能ワークステーション P E R Q 上に実現し、その評価を行った。P r o l o g 処理系をビットマップディスプレイを有するワークステーション上に実現したことにより、マルチウィンドウ等による高品質のマンマシン・インターフェイスを提供できるので本処理系は個人向きの知識情報処理システムを構成する上で極めて強力なツールとなりうる。

### 2. ワークステーション P E R Q

ここではワークステーション P E R Q ( P E R Q 1 A ) についてその概要を説明する。

#### 2.1 C P U

マイクロプログラム制御によるCPUのデータバス幅は16 bit, アドレスバス幅は20 bitである。マイクロ命令は1語48 bitの水平型で1マイクロサイクルは170ナノ秒で16 K語の書き替え可能な制御記憶(W.C.S.)を備えている。基本的にはPERQはUCSD-PascalのP-Codeに類似したPascal向きに設計された中間コード(Q-Code)をマイクロプログラムで直接解釈実行するPascalマシンである。また、このQ-Codeの実行速度は約1 MIPSとなっている。

## 2.2 主記憶と補助記憶

主記憶はアクセスタイムが680ナノ秒(4マイクロサイクル)のRAMを用いて構成され1 MByteの容量を有する。補助記憶としては24 MByteのウインチェスターディスクが用意されている。

## 2.3 オペレーティングシステム

オペレーティングシステムはPOS (Perq Operating System) と呼ばれ、Pascalで記述されている。POSはシングルユーザ、マルチプロセスのシステムであり仮想記憶(仮想記憶空間は4 GByte)、マルチウインドウ等をサポートしており、ファイルシステムは階層化ディレクトリを持っている。またOSの機能はPascal

プログラムから呼び出すことが可能である。

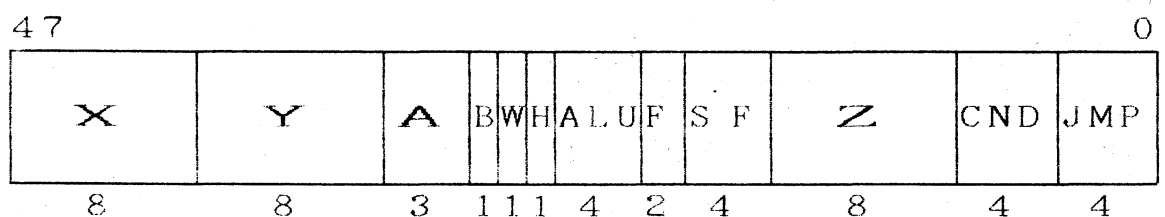
## 2.4 ディスプレイとポインティングデバイス

768×1024ドットの高解像度縦型モノクロディスプレイとポインティングデバイスとしてのデジタイザが用意されており、質の高いユーザインタフェースを与えている。

## 2.5 マイクロ命令

P E R Q の C P U はデータバス幅が16 bitであることから16 bitマシンであるといえるが、アドレスバス幅は20 bitであり1 MByteの記憶空間を扱うことができる。この20 bitのアドレスを計算するためにC P U 内部のレジスタファイル、A L U などのエレメントは20 bitの幅を持っている。

P E R Q のマイクロ命令は1語48 bitの水平型であり各語は図1に示すような12個のフィールドより構成されている。それぞれのフィールドは以下に示すような意味を持っている。



( 数字はビット幅を表す。 )

図1 . マイクロ命令の構成

X : X Y レジスタの X ポートのアドレスまたは X Y レジスタに書き込みを行う場合のアドレスを示す。

Y : X Y レジスタの Y ポートのアドレスまたは定数の下位バイトを指定する。

A : A L U の A ポートの入力を選択する。

B : A L U の B ポートの入力を選択する。

W : X Y レジスタに書き込みを行うか否かを指定する。

H : I O デバイスがメモリーをアクセスするのを禁止する。

A L U : A L U が実行するファンクションを指定する。

F : S F 及び Z フィールドの機能を指定する。

S F : 特殊機能, メモリー制御, ロングジャンプを指定する。

Z : 定数の上位バイト, ショートジャンプ, シフトの制御を指定する。

C N D : 条件付きジャンプの条件を指定する。

J M P : A M D 2 9 1 0 に与える命令を指定する。

### 3 . P r o l o g 処理系の設計

ここでは実現した P r o l o g 処理系の、言語仕様、処理系の構成及び P r o l o g マシンのアーキテクチャについて述べる。なお、以下ではここで構成する P r o l o g 処理系及び言語を P R Q L O G ( P e r q P r o l o g ) と呼ぶ。

#### 3 . 1 言語仕様

P R Q L O G の文法は、現在の P r o l o g の標準となっている D e c 1 0 - P r o l o g に準拠している。すなわち

ホーン節は以下のように記述される。

( F a c t )        h .

( A x i o m )     h : - b<sub>1</sub>, . . . , b<sub>n</sub>.

( D i r e c t i v e )     ? - q<sub>1</sub>, . . . , q<sub>n</sub>.

例 .

a p p e n d ( [ ] , X , X ) .

a p p e n d ( [ A | D ] , X , [ A | Y ] ) : -

a p p e n d ( D , X , Y ) .

? - a p p e n d ( [ a , b , c ] , [ d , e ] , X ) .

### 3.2 処理系の構成

P E R Q は P a s c a l マシンであり、その O S も P a s c a l で記述されている。従って処理系が O S の機能（メモリーアロケーション、I/Oハンドリング等）を利用するためには P a s c a l マシンと P r o l o g マシンを共存させる必要がある。このため、システムは P a s c a l で記述されている部分とマイクロプログラムで記述されている部分から構成されている。この両者はお互いにコールし合いながら P r o l o g プログラムを実行する。

P a s c a l で記述されている部分はシステム・スーパーバイザ及びマイクロプログラム・レベルでは記述が極めて煩雑となる入出力を扱う部分である。システム・スーパーバイザは

マイクロプログラムのロード、呼び出し、メモリーの割り当て等を行う。入力ルーチンは項（P r o l o g プログラム等）を読み込み、構文解析を行いマイクロプログラムルーチンのための内部表現に変換し、出力ルーチンはこれと逆に内部表現を文字列に変換し出力する。

マイクロプログラムで記述されている部分は P r o l o g インタプリタで主要となる部分であり、ユニフィケーション、バックトラッキングなどのアルゴリズムが主体となる特に高速性が要求される部分である。図2にこの考え方に基づいた P R Q L O G の処理系の構成を示す。この図における上下関係は依存関係を示す。また P E R Q のレジスタおよび W . C . S . の割り当てを図3に示す。

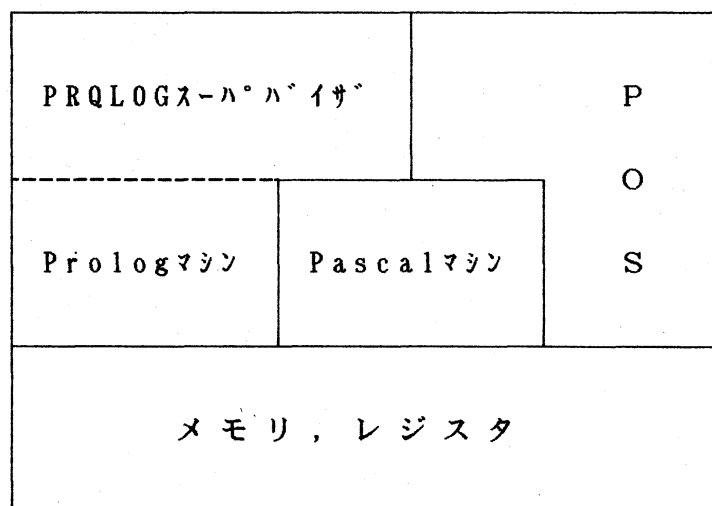


図2 . P R Q L O G 処理系の構成

図3に示されているように P R Q L O G 処理系（P r o l

o g マシン) のための領域は P a s c a l マシンの領域とは独立に確保されている。

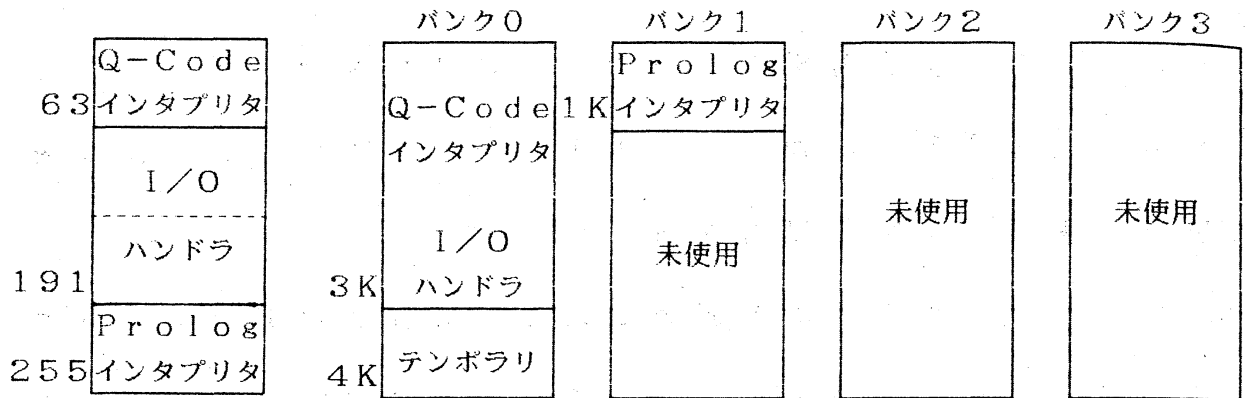


図3. レジスタおよび W.C.S. の割り当て

### 3.3 Prolog マシンのアーキテクチャ

Prolog マシンは以下の7つの要素により構成されている。

- (i) 項領域：節の内部表現の基本単位である項を格納する。
- (ii) シンボルテーブル：シンボルを管理する。
- (iii) 述語テーブル：述語の登録を管理する。
- (iv) エンバイロメントスタック：証明手続き中の環境を保存する。
- (v) ユニフィケーションスタック：構造体の再帰的なユニファイを実現する。
- (vi) トレールスタック：バックトラックの際に変数の束縛を解放するのに用いる。
- (vii) レジスタ群：テーブル、スタック等へのポインタを格納



する。

(i)～(vii)を実現するために様々なデータ構造が用いられる。次にこれらのデータ構造について述べる。

P E R Q は 1 6 bitマシンであるのでデータの内部表現も 1 6 bitを基本単位としている。また P E R Q の仮想記憶はセグメンテーション方式であり 1 つのデータセグメントの大きさは最大でも 1 6 bitでアドレス可能な大きさ ( 1 2 8 KByte ) である。このため内部表現で用いるアドレス ( ポインタ ) はセグメント内でのオフセットアドレスを用い 1 6 bitで表現することとする。

#### ( 1 ) 項の内部表現

節の内部表現における基本単位は項と呼ばれるものである。項は図 4 に示すような 3 2 bitの項記述子によって表現される。

型	値
---	---

項記述子は 1 6 bit× 2 語で構成

され、第 1 語は型フィールドでそ

の項の種類を示し、整数、シンボ

ル、変数、構造体に分類する。第

2 語は値フィールドで項の値を示す。

各々の型に対する値フィールドの内容は以下のようなものである。

整数 : 値は整数値の 2 の補数表現。

シンボル : シンボルを表すシンボル記述子のアドレス。この

アドレスはシンボルの識別子として用いられる。  
すなわち、シンボルは識別子が等しい時またその  
時に限って同一であると見なされる。シンボル記  
述子については後述する。

変数 : 節中でその変数が何番目に現れた変数であるかを  
示す番号。例として、

$a(X) : -b(Z, X), c(Y)$  . という  
節を考えると、変数  $X$  には 1,  $Y$  には 3,  $Z$  には  
2 という番号が付けられる。

構造体 : 構造体を表す構造体記述子のアドレス。構造体記  
述子については次に述べる。

### (2) 構造体記述子

構造体記述子は構造体を表現するのに用いられる。構造体は  
構造体名と 1 つ以上の引数を持つ。構造体記述子は引数の数  
及び構造体名を表すヘッダと引数を表す項記述子からなる。  
引数の数は 16 bit で表され、構造体名はシンボル記述子のア  
ドレスとして表される。また各引数を表す項記述子がヘッダ  
の次に格納される。

### (3) シンボルテーブル

システム内に存在する全てのシンボルは全てそれに対応す  
るシンボル記述子により識別される。シンボル記述子はオー

アンハッシュ法によりハッシュテーブルとして構成されているシンボルテーブルにより管理される。

シンボル記述子は次の5つのフィールドを持つ。

(i) シンボルを表す文字列の長さ：8 bitの整数。

(ii) シンボルの持つ属性：属性を8 bit表現したものであ

り、属性には Prefix, Infix, Suffix, Othersの4つがある。

(iii) 述語記述子のアドレス：もしそのシンボルを用いて述語が構成されていれば述語記述子のアドレスを16 bitで格納する。

(iv) ハッシュリンクポインタ：同じハッシュ値を持つシンボル記述子に対するリンクポインタ。

(v) シンボルを表す文字列：シンボルを表す文字列を8 bitのASCIIコード列で表したもの。

#### (4) 述語テーブル

述語テーブルは、節のヘッドリテラルに用いられているシンボルをキーとして構成された述語の表である。シンボルが与えられるとそれに対応するシンボル記述子を用いて述語記述子のアドレスを知ることができる。図5に述語テーブル及び述語記述子の構成を示す。



子へのポインタ。

(iv)項記述子，サブルーチン番号またはマイクロアドレス：

その述語が P r o l o g で実現されている場合には  
項記述子が、P a s c a l の場合にはサブルーチン  
番号が、マイクロプログラムの場合にはマイクロア  
ドレスが格納される。

図5の述語テーブルの状態はあるシンボルを用いて2引数の述語が3つ，0引数の述語が2つ，1引数の述語が1つ登録されている状態を示している。

#### (5) エンバイロメントスタック

述語テーブルに登録された節 ( F a c t , D i r e c t i v e ) が証明のために用いられると環境がエンバイロメントスタック上に作られる。環境には次の6つの情報が含まれる。

(i)ゴール：その節を呼び出した項記述子のポインタ。

(ii)親の環境：その節を呼び出した節の持つ環境。

(iii)トレールポインタ：その節の呼び出しの時点のトレールスタックポインタの値。

(iv)バックトラックポイント：ゴールが失敗した場合にバックトラックする環境のポインタ。

(v)次候補節集合：その節の適用に失敗した場合に用いる節のリスト。

(vi) バインドリスト：変数のバインドを表すセル。

以上のように原始項と変数の束縛からなるインスタンスの表現にはストラクチャシェアリング方式を用いた。

#### 4. 処理系の動作

処理系に入力された文字列は初めにトークンと呼ばれる単位に切り分けられ先読みのためのキューに置かれる。パーザはこのキューを参照してトップダウンパーズングを行い、項記述子を項領域上に作る。入力された節が Directive の場合には P a s c a l マシンが P r o l o g マシンを呼び出す。P r o l o g マシンは項領域、シンボルテーブル、述語テーブルを参照し、適用した節に対して環境をエンバイロメントスタックにプッシュしていく。構造体のユニファイの際には再帰的ユニファイの実現のためにユニフィケーションスタックが用いられる。束縛の起こった変数セルのアドレスはトレールスタックにプッシュされバックトラックが起こった場合にはこのスタックを参照して束縛を解放する。P a s c a l で実現された述語の呼び出しに対しては P r o l o g マシンが逆に P a s c a l マシンを呼び出す。ゴールが全て成功した場合には制御が P r o l o g マシンから P a s c a l マシンへ戻りトップレベルの変数の値を出力して入力待ちになる。

## 5. 処理系の評価

P r o l o g 処理系は一般に次のような点から評価される。

### (i) 性能に関するもの

処理速度，ユーザーが使用可能なメモリー領域

処理速度の評価については W a r r e n らの用いた 30 要素のリストのナイーブリバースを用いた。この実行には約 0.1 秒を要し、これは約 5 K L I P S に相当している。また、ユーザーが使用可能なメモリー領域は項領域，述語テーブル等のヒープ領域が約 1 2 8 KByte、エンバイロメントスタックのスタック領域が 1 2 8 KByte の合計 2 5 6 KByte である。

### (ii) 言語機能に関するもの

シンタックス，グラフィック機能，コンパイラ

本処理系のシンタックスは D e c 1 0 - P r o l o g に準拠しており、グラフィック機能については将来ビットマップディスプレイを活かした述語を取り入れる予定である。またコンパイラについては P r o l o g 向きに設計された中間言語をオブジェクトとするコンパイラを P r o l o g で記述しその中間言語をマイクロプログラムで直接解釈実行する方式を計画している。

### (iii) プログラム開発環境に関するもの

### エディタ，デバッガ

現在のところエディタは P E R Q の O S のユーティリティーであるスクリーンエディタを用いている。このエディタは P r o l o g の外から呼び出すかたちになっている。またデバッガはない。

### (iv) 応用

#### 他言語とのインターフェイス

他言語とのインターフェイスは P a s c a l とのインターフェイスを取ることが可能である。

### 6. あとがき

今後の課題としては、組み込み述語の整備，効率的なバックトラックを可能にするなどの処理系の最適化，5. で述べた方式によるコンパイラの作成が考えられる。